3D Model Generation from a Stereo Camera using Multiple Viewpoints

Adrian Clark[1], Richard Green[2]

University of Canterbury, Department of Computer Science
Private Bag 4800, Christchurch, New Zealand
[1] adrian.clark@canterbury.ac.nz
[2] richard.green@canterbury.ac.nz

Abstract

This paper explores some existing methods commonly used in three dimensional (3D) model generation, in particular those which typically employ multiple cameras or multiple frames from one camera for image acquisition. A novel method of model generation is described and evaluated. This method captures multiple images from different viewpoints with a stereo camera to create separate three dimensional meshes, which are then joined to form one single continuous mesh which represents the captured object in 3D. The results from the method provide a good base for 3D models, and although the quality of the results may not be as high as other more expensive methods, they can be achieved at low cost and with minimal set up time and calibration.

Keywords: 3d Model Generation, Stereo Camera

1 Introduction

Many applications in multimedia and entertainment are moving towards three-dimensional graphics. One only needs to take a look at the commercial computer and console game market to see that the trend of multimedia is towards creating a realistic and immersive experience. Humans visually perceive the world in three dimensions, and thus to make multimedia as realistic and immersive as possible, three dimensional graphics are necessary. However, a major draw back of 3D graphics is the time consuming process of object creation, where real world objects must laboriously be described in a way that computers can understand and represent them.

This paper describes a method of cheap and easy generation of models with a minimal emphasis on human-computer interaction, so that even users without knowledge of modelling tools can generate 3D content. With the use of a simple stereo camera targeted at the object to be modelled, users can generate an approximate mesh representing the object, which can be used as is, or refined for greater definition. The algorithm used by the stereo cameras is a simple disparity calculation, and is described in further detail in the Image Acquisition section.

2 Background

There are three main approaches to creating three-dimensional content. The first is simply creating all content manually using computer based modelling tools; however the major drawback of this is that it takes considerable skill and time. There are methods that can aid the user in manual creation. One such method is to use existing photographs as a base, then use "three-dimensional parametric primitives, such as cubes, cylinders, cones etc" [1] to map the scene to objects. While this does aid the user somewhat, it still requires a considerable amount of labour intensive work, and requires some knowledge of modelling tools, which are often complex and expensive.

The second method is to use a three-dimensional scanner, typically using some form of laser. These scanning lasers provide a "high-precision, high-density three-dimensional (3D) reconstruction" [2] of the object. However, these lasers are also expensive, and as such, may not be easily available to novices who wish to create 3D content. There is also a certain amount of work involved the set-up and capture of objects using laser scanners, including "planning a set of views, physically altering the relative object-sensor pose, taking scans, registering the acquired geometric data in a common coordinate frame of reference, and finally integrating range images into a non-redundant model." [2]. While laser scanning does provide the best results in regards to accuracy, it's prohibitive cost and set-up time means this method may not be practical for small companies and amateur users.

However, there are companies who work towards making this type of scanning available to the common market. One

such company is the Christchurch based ARANZ [3]. One product they have developed is a hand-held laser scanner, known as the Polhemus FastSCAN [4], which was used by companies such as Weta Digital for the 3D scanning and modelling of the creatures used in The Lord of the Rings films. This scanner has also been used for medical purposes, such as the creation of prosthetics and forensics.

The final method for object creation draws upon computer vision techniques. Of these, there are typically two methods for extraction of 3D information: motion capture (multiple frames from a single camera) and analysing the differences between pictures taken of an object from varying locations (multiple cameras). While motion capture no longer requires expensive image capturing hardware (a $100 USB2 camera and standard PC will do), it does require computationally expensive algorithms, and relies on specific domain knowledge, such as "the relative motion of the camera between the two images" [5]. Generating 3D models using motion capture can be optimised by special equipment, such as a fixed speed turntable, or some sort of rail system that the camera can run on. This allows the camera's path and speed to be maintained at a known value.

Finding disparity between two images using multiple cameras on the other hand only requires the object and two images taken of it from different locations. This can easily be captured by equipment such as a stereo camera, and processed by image processing libraries such as the SRI Small Vision System [6]. For every point that can be registered between the two images, a distance can be calculated from the comparison of two images, which provides information about the 3D structure of an object relative to the camera. This means that that only 3D structure information about the current side of the object which is facing the camera is obtained, rather than about the object as a whole.

3 Solution

From the approaches mentioned above, the disparity calculation using stereo cameras fits the requirement of a low cost and computationally efficient method of 3D model generation well. However, as mentioned this method only provides information about a model from one vantage point, which is what is visible from camera location at the time. The research performed was to combine images taken from multiple cameras with different viewpoints around the object. This would gather enough information which, when combined would provide a complete 3D model of the object.

The method used from image acquisition to model generation is described in the steps below.

3.1 Capture Set-up

The set-up consisted of a computer running Windows XP, connected to a Stereo Camera. The stereo camera was set up with a focal length of approximately 3 metres. The subject (in this case a person holding an object in front of them) was placed in front of a green screen, at the focal length of the camera, on a rotating turntable. An image was taken of the Front, Back, Left and Right sides of the object, using the SRI Small Vision System. Figure 1 shows a two-dimensional capture from one of the captures, showing the subject to be captured (in the front view).

3.2 Image Acquisition

The stereo camera uses the difference in location of points in two pictures to calculate the distance the object is from the camera. Each camera takes a separate picture of the object, and then these two images are compared to find the amount of difference between pixels, to calculate the distance each pixel is from the cameras.

Figure 1: The Subject Capture Set-up

Figure 2 is show in the SRI small vision systems user manual [7], and shows how disparity is calculated. The point for disparity calculation is the point that lays on the hexagonal object's edge closest to the cameras, which are shown as the black dots. The distance between the cameras is called the baseline, *b*, and the focal length of the lenses is *f*. Using the formula below, the distance the point is from the cameras, *r*, can be calculated as:

$$r = bf / d \quad \text{, Where } d = dl - dr \tag{1}$$

In this formula, d is the disparity between the point on the left and the right image. For every point that can be identified in both the left and right images, this formula can be used to calculate its depth.

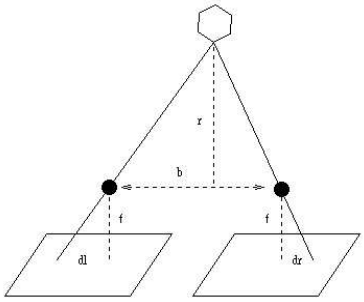

Figure 2: Calculation of Disparity

This research aims to build a model of the entire object; where as a single capture only provides information about one side of an object. Because of this, there needs to be multiple shots taken in this way from around the object. The software which was designed for this project only used four views from the Left, Right, Front and Back of the object, though through extensions this software could take any number of images from around the object and, as long as the location of the camera in regards to the centre of rotation of the object is recorded, combine all these meshes to get a more accurate representation of the object.

Figure 3 shows the Left and Front views of the object as a two-dimensional screenshot. The areas which are shown as white are areas which the disparity calculation was unable to calculate a depth value for. This is usually because of areas of indistinguishable colours. The algorithm is unable to find any matching points in these areas between the two images, and as such cannot calculate the disparity of the points. This is a major problem when dealing with stereo cameras. As the stereo camera was unable to calculate depth values for major areas, such as in the torso and legs, these areas cannot be precisely modelling, and instead there must be some kind of interpolation performed such that the model does not have gaps appearing. These interpolation methods are discussed in the Construction of Meshes Section

Figure 3: The pixels that have computed depth values

3.3 Construction of Meshes

The data returned from the stereo cam using the SRI Small Vision System is a file that contains a number of entries, one for each pixel. The file contains each pixel's X, Y and Z co-ordinates, as well as the colour of the pixel at that point. The Small Vision System allows background thresholding, which allows the removal of the background from the image. This is recorded with a negative value for the Z co-ordinate. All the pixels that the system was not able to determine a Z co-ordinate value for, usually because of problems identifying the pixels in each source image, are also given a negative value for the Z co-ordinate. Because these negative values are points which are either part of the background or unusable due to not having enough data to represent them, these values are simply ignored by the algorithm.

Figure 4 shows two views of the four point clouds in 3D combined together. Already the final model is beginning to take shape. However, the gaps that were shown as white spaces in Figure 3 are once again present, and a good deal of the legs of the subject are missing. To deal with this problem, the algorithm used a basic linear interpolation and seeking function to create the meshes.

Figure 4: The Points in 3D from all four images

Because Meshes are built from triangles, for each point in the mesh there needs to be two other neighbouring points known. For each pixel, the program stores the index of the pixel to its right and below it, so that when the mesh is created, it can use those three points as the vertices of a triangle. This is where the first significant problem arose: pixels whose neighbouring pixels had no value, due to the fact that they were part of the background or there was not enough data available to calculate a Z co-ordinate value for them. This left gaps in the mesh, like those shown in Figures 3 and 4. To resolve this, a linked list was used to store every visited point and its predicted right neighbour and bottom neighbour. For every point *p*, the list was traversed to find all the points (*p1, p2…*) that consider it a neighbour. If the point *p* had no value, point's *p1, p2…* would note that this point didn't exist, and choose the next point as the new neighbour. If however the point *p* was a valid point, point's *p1, p2…* were stored in the mesh with this neighbour confirmed. As is shown in Figure 5, any gaps that occur in the mesh (shown by non-filled circles) will be covered by a simple linear interpolation (shown by the grey lines). This is an efficient solution, as it simply selects the next point from left to right, top to bottom, as the next neighbour (a better method is discussed in the Further Work section), but in the source images which was used, this method performed surprisingly well.

Figure 5 shows an interpretation of a point cloud. To find the right and bottom neighbours of point *A*, the method first assumes that they are next to the point itself, and adds them to the list as such. When point *B* is examined, it is found that it has no value, so point *A*'s right neighbour is updated to the next point, point *C*. When point *C* is reached, it is found, and point *A*'s right neighbour is confirmed to be point *C*. The same process is performed for the bottom neighbour, which is predicted to be point D, until point D is found to be missing, when the neighbour is updated to point E.

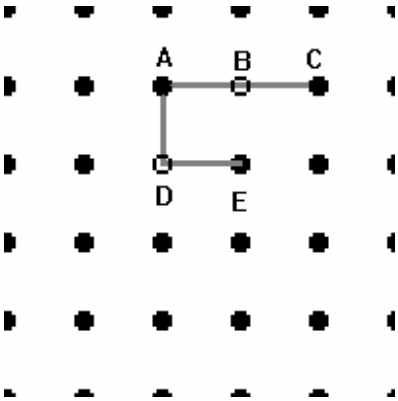

Figure 5: The mesh joining method

One side of a completed mesh is shown in Figure 6. As can be seen, occlusion is a problem here. As the stereo camera can only see objects from one direction, it cannot make any assumptions about objects occluding one another. In this example, the object being held by the subject occludes part of his body. The stereo camera has no way of knowing the object held is not part of his body, and because of this the object is connected to his body in the mesh. This is not an easily fixable problem; the simplest method would be to separate the points into groups based on depth values, and then perform mesh construction on each group. In this example, the subject's body would be one group, and the arms and held object another, this would cause the body to be considered a separate entity from the arms and object, and there would be no joins between them.

Figure 6: The single front mesh, showing the occlusion problem

3.4 Mesh Combination and Model Generation

Now that there is a mesh for each side of the object, it is simply a matter of translating all the meshes into their respective locations, and combining them. In the implementation used, there is only a mesh for each of the four sides of the object, which means there will be little overlap. However, for more complex implementations of this method, we can use a larger number of meshes generated from around the object. This means that certain areas of the object will be represented by more than one mesh. This can be used to an advantage. Computer vision techniques often suffer from camera noise, and taking multiple samples and averaging them reduced the effects of this noise. For this, the meshes should be overlaid onto one other. For each vertex common to two or more meshes, the distance value for that vertex from each mesh can be added together, and then the resulting valued divided by the number of meshes sampled. This has the effect of smoothing out the mesh, as well as removing any overlap between meshes.

Another problem that can be corrected in this final step is gaps between the meshes. If there is incorrect calibration of the camera or object - possibly the distance between the camera and the objects centre of rotation has changed between shots, the two meshes may not meet up correctly. When combining these meshes there will be a gap between them. The easiest way to deal with this is just to do a straight edge vertex to edge vertex join across two meshes. However, this will cause angular edges at the point of every edge join, which may not be desirable. In this case, there was a need for interpolation between edges. Using the formula for Hermite Interpolation:

$$p = (2t^3 - 3t^2 + 1)p_1 + (t^3 - 2t^2 + t)m_1 + (-2t^3 + 3t^2)p_2 + (t^3 - t^2)m_2 \quad (2)$$

For both meshes, as long as there is an edge point (p1, p2) known and the gradients at each of those points (m1, m2), the formula can produce any number of values of t between 0 and 1 exclusive, to generate new vertices which will produce a smooth edge between the meshes. Once these steps are performed, the final mesh is available.

4 Results

Once the images were put into the program for computation, the model shown in Figure 7 was generated.

4.1 The output model

As shown in Figure 7, the model is a close representation of the original image, and therefore it does bear a considerable resemblance to it. Depending on how much detail is required of the models, they can either be used as is, or perform manual manipulation of the mesh to increase the accuracy compared with the original model. Either way, a

considerable amount of work has been saved compared to just creating the model from scratch.

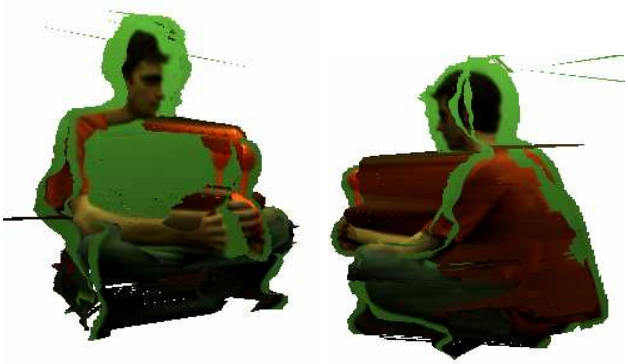

Figure 7: The Meshes from all four images combined

One apparent problem with the model is the green edging around the various mesh parts. This was a problem caused by the stereo camera, which encountered problems computing the depth values of the background close to the subject. While some of it can be removed from the image by changing the background threshold value, most of the points in that area have the wrong depth value and have a considerably smaller Z co-ordinate value than they should. To resolve this requires an additional processing step in the model creation stage, which can use either Hue subtraction or Edge Removal.

4.2 Hue Subtraction

For hue subtraction, the user would need to state which hue the background was (in this case, a green). All points with this hue value would be disregarded, as were the points that the camera couldn't determine a depth value for. While this is computationally simple, it does mean that the background surrounding the object must be a single hue to work properly. Another negative feature is that if the subject contains any points with the hue of the background, they would also be subtracted.

It was decided to implement this algorithm, to examine its performance. As is shown in Figure 8, the results were surprisingly good. As the subject was placed in front of a plain green background, and was not wearing any green clothing, the hue subtraction was able to remove all traces of the background without interfering too much with the actual subject. However, although the subject was wearing blue pants, the algorithm detected some instances of the same green hue as the background contained in them, and removed parts of the pants.

4.3 Edge Removal

Edge removal consists of finding the edges around a mesh, and then removing them. For this, the algorithm needs only to find points that have no neighbours on one side and remove them. Performing this action a number of times, the edges around the object will gradually disappear. However this required intervention from the user, and was more computationally intensive.

Figure 8: The model after background Hue Subtraction

5 Conclusions

The aim of the software developed was to develop a computationally efficient, low cost method of generating three-dimensional models of existing objects for use in Multimedia and entertainment applications. The research described users a stereo camera for all image capturing, and results were processed offline to generate the model. While the models are not as accurate as might be obtained from other methods such as ARANZ's laser scanning methods, the low-resolution 3D model is usable when the required detail for content is low, and can be used as a basis for further refinement of the model to achieve greater detail levels.

There is much room for improvement in this method before it becomes a viable means of model production. Even with such an efficient method, the results show that there may be some merit in performing model capture this way. Some of the ideas and problems that have been encountered during this research are described below, as well as further research directions to explore these problems, and refine the proposed model.

5.1 Further Work

The first option this project would benefit from is an increased number of source images. Currently the method uses four images from the four main sides of an object. However, for higher resolution images, it is conceivable that a number of images could be taken with closer proximity to the object, and then image mosaicing could be used to integrate these smaller parts into a larger mesh. Complex objects would also benefit from more images taken from smaller angle increments around the object.

As mentioned in the Construction of Meshes section, there is a problem creating meshes from the 3D point cloud when there are missing points. The method described uses an efficient linear method, which just keeps looking iteratively for the next neighbouring point. A more accurate method

would load the mesh into a two-dimensional array, and for each point perform an outward scan, checking for existing points. All returned points could then be given a score, depending on the vector between the source point and the found point, compared to the vector between the source point and its ideal neighbour (i.e. the point directly beside this point). The point with the smallest difference would be chosen. This would reduce the errors in the mesh when covering gaps.

Another research direction for this problem would be to extend the Hermite Interpolation formula mentioned in the Mesh Combination and Model Generation section. Using the start of a gap and the end of a gap as the points, and the gradients at these points in the direction of the gap, the t value could be varied to interpolate what the values of the missing points would be. Performing this extra step would minimize the need for making the meshes stretch across the gaps, as most gaps could be filled by this method.

The mesh creation step also has a problem in regards to occlusion. When one object is in front of another the stereo camera has no ability to tell if they are part of the same object or not. Earlier in this paper it was suggested that meshes could be broken into groups of points based on their Z co-ordinate values. In the example used in the paper of the subject holding an object in front of him, this would separate the subject and the object into two distinct entities, which would then be generated into two meshes.

Another research direction which could be pursued to resolve this problem would be to have an intermediate step using Voxels. A Voxel is a volumetric pixel, and is essential a pixel with an X, Y and Z value in space. If meshes are thought of as box kites, with a frameworks of edges joined with fabric (or in the graphics world, triangles), then voxels are similar to Lego, where a complete object is made of lots of smaller blocks. A good approach for this method would be one similar to Ken Silverman's [8] method for voxel creation. First a cube is created in the maximum size final model, using the example this project does of four 320 by 240 images, a cube of size 320 by 320 would be created. Using the Z values from each pixel of each of our source images, voxels can then be removed from the solid cube, until there is a voxel representation of the object, which can then be converted into a mesh. The major drawback with this is that, since the cube is of a fixed size, all our depth values must be scaled to fit within the cube (e.g. such that the minimum Z value is 0 and maximum is 320), which means that precision may be lost in the models.

6 References

[1] S. Gibson, T. Howard — Interactive Reconstruction of Virtual Environments from Photographs, with Application to Scene-of-Crime analysis. Proceedings of the ACM symposium on Virtual reality software and technology (2000), pp 41-48.

[2] W. R. Scott, G. Roth — View planning for automated three-dimensional object reconstruction and inspection. ACM Computing Surveys (March 2003), pp 64-96.

[3] ARANZ Company Profile — http://aranz.com/about/

[4] Polhemus FastSCAN — http://www.fastscan3d.com/

[5] M. Pollefeys, L. Van Gool — From Images to 3D Models. Communications of the ACM (July 2002), pp 50-55.

[6] SRI Stereo Engine — http://www.ai.sri.com/~konolige/svs/

[7] K. Konolige, D. Beymer — SRI Small Vision System: User Manual. http://www.videredesign.com/smallv-3.2.pdf

[8] Ken Silverman's Voxlap Page — http://advsys.net/ken/voxlap.htm